Organisasi dan Arsitektur Komputer KJ002

CACHE MEMORY PADA ORGANISASI DAN ARSITEKTUR KOMPUTER

Disusun oleh

Nama : Marcel Immanuel

NIM : 20240801117

Mata Pelajaran : Organisasi dan Arsitektur Komputer

Nama Dosen : Dr. BUDI TJAHJONO , S.Kom, M.Kom

TAHUN PELAJARAN 2025/2026

PEMERINTAH PROVINSI DAERAH KHUSUS

IBUKOTA JAKARTA DINAS PENDIDIKAN

WILAYAH 1 JAKARTA BARAT UNIVERSITAS

ESA UNGGUL

Jl. Arjuna Utara No.9, Duri Kepa, Kec. Kb. Jeruk, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta 1151

Abstract

Memory hierarchy plays an important role in optimizing memory retrieval and overall system efficiency. The memory hierarchy categorizes computer storage based on speed, complexity, capacity, performance, and technology used. Understanding the different types of memory, such as cache memory, registers, and main memory, is crucial to improving computer system performance. Cache memory, including L1 and L2 cache, provides fast data access to the processor, thus reducing access time for previously accessed data without the need to perform a move. Although the L2 cache is larger than the L1 cache, operating at a slower speed with higher latency, its presence is not mandatory in all processors. Registers such as the DS register specifies the segment where program data is stored, usually requiring little change unless it is related to a program that requires changes. In addition, there is the challenge of balancing cache capacity and performance, as larger cache sizes require more gates for management, potentially impacting system efficiency. Different types of memory technologies, such as DRAM and SRAM, offer various trade-offs between cost, speed, and refresh requirements.

Keywords: *Memory hierarchy, cache, main memory, external memory, Computer architecture*

Abstrak

Hirarki memori memainkan peran penting dalam mengoptimalkan pengambilan memori dan efisiensi sistem secara keseluruhan. Hirarki memori mengkategorikan penyimpanan komputer berdasarkan kecepatan, kompleksitas, kapasitas, kinerja, dan teknologi yang digunakan. Memahami berbagai jenis memori, seperti memori cache, register, dan memori utama, sangat penting untuk meningkatkan kinerja sistem komputer. Memori cache, termasuk cache L1 dan L2, menyediakan akses data yang cepat ke prosesor, sehingga mengurangi waktu akses untuk data yang telah diakses sebelumnya tanpa perlu melakukan pemindahan. Meskipun cache L2 lebih besar daripada cache L1, beroperasi pada kecepatan yang lebih lambat dengan latensi yang lebih tinggi, keberadaannya tidak wajib di semua prosesor. Register seperti register DS menentukan segmen di mana data program disimpan, biasanya hanya membutuhkan sedikit perubahan kecuali terkait dengan program yang memerlukan perubahan. Selain itu, terdapat tantangan untuk menyeimbangkan kapasitas dan kinerja cache, karena ukuran cache yang lebih besar membutuhkan lebih banyak gerbang untuk manajemen, yang berpotensi berdampak pada efisiensi sistem. Berbagai jenis teknologi memori, seperti DRAM dan SRAM, menawarkan berbagai trade-off antara biaya, kecepatan, dan kebutuhan penyegaran.

Kata Kunci: Hierarki Memori, Cache, Memori Utama, Memori Eksternal, Arsitektur Komputer.

1. PENDAHULUAN

Memori merupakan salah satu komponen kunci dalam arsitektur komputer yang memainkan peran vital dalam menyimpan dan mengakses data dengan efisien. Dalam analisis struktur memori dalam arsitektur komputer, hierarki memori memegang peranan penting dalam menentukan kinerja sistem secara keseluruhan. Dengan pemahaman yang mendalam tentang berbagai jenis memori, mulai dari register hingga memori sekunder, serta faktor-faktor seperti kecepatan, kapasitas, dan biaya per bit, kita dapat merancang sistem memori yang optimal untuk kebutuhan komputasi modern.

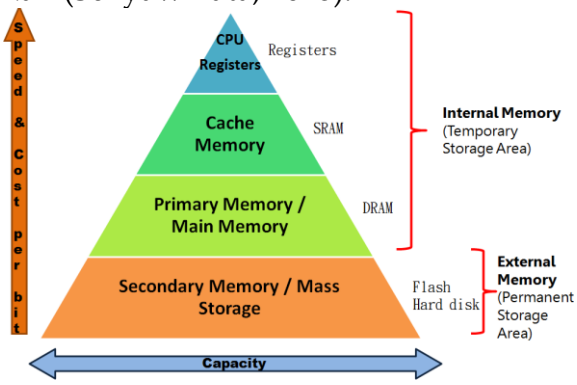
Dalam konteks penelitian ini dilakukan melalui metode kajian literatur, yang memberikan wawasan yang berharga tentang struktur memori dan hierarki memori dalam arsitektur komputer. Dengan melibatkan analisis komparatif terhadap berbagai jenis memori, seperti DRAM dan SRAM, serta memori eksternal seperti magnetic disk dan optical disk, kita dapat memahami kelebihan dan kekurangan masing-masing jenis memori.

2. METODOLOGI PENELITIAN

Pada metode penelitian analisis ini, menggunakan metode kajian literatur dengan melibatkan pencarian dan analisis terhadap sumber-sumber teoritis yang relevan mengenai struktur memori dan hierarki memori dalam arsitektur komputer. Dalam analisis literatur, peneliti dapat mengumpulkan informasi dari berbagai referensi seperti buku, jurnal, dan artikel terkait. Dan juga menggunakan analisis Komparatif dengan Membandingkan berbagai jenis memori seperti register, cache, memori utama, dan memori sekunder untuk memahami kelebihan dan kekurangannya dalam konteks penggunaan dalam sistem komputer.

3. HASIL DAN PEMBAHASAN

Hierarki memori dalam arsitektur komputer mengklasifikasikan penyimpanan komputer berdasarkan kecepatan media penyimpanan, serta faktor-faktor seperti kompleksitas, kapasitas penyimpanan, kinerja, dan teknologi yang digunakan (Suryawinata, 2018).



Gambar 3. 1 Ilustrasi Hierarki Memori Sumber ()

Pada gambar (3.1) di atas mengilustrasikan hierarki memori berbagai jenis memori di komputer. Hierarki ini ditentukan oleh faktor-faktor seperti kecepatan, kapasitas, dan biaya per bit. Umumnya, level yang lebih tinggi dalam hierarki lebih cepat namun memiliki kapasitas yang lebih kecil dan biaya per bit yang lebih tinggi, sedangkan level yang lebih rendah lebih lambat namun memiliki kapasitas yang lebih besar dan biaya per bit yang lebih rendah. Dalam

hal ini penulis berusaha mengulas satu persatu susunan hierarkis pada memori ;

1. Register

Register merupakan salah satu komponen memori mikroprosesor, dikenal dengan kecepatan aksesnya yang tinggi dan biaya per-byte yang mahal (Victor Amrizal & Qurrotul Aini, 2013). Register memainkan peran penting dalam membantu mikroprosesor dengan melakukan tugas-tugas yang tidak dapat ditangani langsung oleh prosesor, dan bertindak sebagai anggota tubuh prosesor. Sebelum prosesor dapat memanipulasi data, semua tipe data harus dikenali oleh register (Rima Rizqi Wijayanti, S.ST., 2023). Sebagai contoh, dalam operasi aritmatika, bilangan masukan dan keluaran disimpan dalam register. Register biasanya diukur dalam bit untuk menunjukkan kapasitas datanya, dengan panjang yang paling umum adalah 32-bit dan 64-bit .

Register dibagi menjadi beberapa segmen berbeda, seperti CS (*Code Segment*), DS (*Data Segment*), ES (*Extra Segment*), SS (*Stack Segment*) dengan setiap segmen berisi register 16-bit. Register ini biasanya digunakan untuk menentukan alamat memori suatu segmen.

a) CS (*Code Segment*)

Segmen aktif saat ini dilambangkan dengan register CS (Code Segment), sedangkan segmen yang digunakan oleh stack ditandai dengan register SS (Stack Segment). Sangat penting untuk berhati-hati saat memodifikasi register ini, karena perubahan apa pun yang ceroboh berpotensi menyebabkan gangguan dan komplikasi pada program di kemudian hari.

b) DS (*Data Segment*)

Register DS biasanya digunakan untuk menentukan segmen dimana data program disimpan. Biasanya, tidak perlu mengubah isi register ini kecuali jika berhubungan dengan program residen yang mungkin memerlukan perubahan.

c) ES (*Extra Segment*)

Register ES (Extra Segment) berfungsi sebagai register tambahan tanpa fungsi khusus yang ditugaskan padanya. Biasanya digunakan untuk menunjuk ke alamat memori, seperti lokasi di memori video. Pada prosesor 80386 juga terdapat dua register segmen 16-bit lagi yang dikenal sebagai FS dan GS. Register ini memberikan fleksibilitas dan fungsionalitas tambahan untuk pengalamatan memori di prosesor.

Register juga memiliki pointer dan index register seperti SP (*Stack Pointer*), BP (*Base Pointer*), SI (*Source Index*), dan DI (Destination Index), masing-masing panjangnya 16 bit. Register ini digunakan untuk menunjuk ke lokasi tertentu di memori, seperti register SP yang dipasangkan dengan register segmen SS untuk menunjuk ke alamat tumpukan, dan register BP yang dipasangkan dengan register SS untuk menunjuk ke alamat memori tempat data disimpan (Munazilin, 2017). Register SI dan DI sering digunakan dalam operasi string untuk mengakses alamat memori secara langsung. Selain itu, register general purpose seperti AX, BX, CX, dan DX, yang terdiri dari 16 bit, digunakan untuk menyimpan sementara data yang sedang dimanipulasi oleh CPU. Register special purpose memiliki kemampuan untuk menyimpan instruksi yang menyimpan alamat pengirim untuk operasi selanjutnya yang akan dilakukan oleh prosesor.

2. Cache Memory

Cache, diucapkan sebagai "kaesh", berfungsi sebagai lokasi penyimpanan sementara yang membantu mempercepat proses pengaksesan data dengan menyimpannya secara berkala untuk pengambilan cepat bila diperlukan (Abdurohman, 2017). Dalam arsitektur komputer, cache adalah jenis memori yang berada di bagian atas hierarki memori, tepat di bawah register. Meskipun cache memiliki kapasitas terbatas, efisiensinya sebanding dengan prosesor.

1) Tingkatan *Cache Memory*

Cache disusun berdasarkan kedekatannya dengan prosessor, susunan ini menggunakan level seperti L1 (Level 1), L2 (Level 2), L3 (Level 3) (Putri, 2008). Sedangkan dalam ukurannya Cache dinyatakan dalam satuan KB (KiloByte). Misalkan 6KB, 12KB, 54KB, dan seterusnya. Memori cache dikelaskan ke dalam tingkatannya sendiri:

a) L1 (Level 1)

Cache L1 adalah jenis memori cache yang terintegrasi langsung ke dalam prosesor. Ini digunakan untuk menyimpan informasi yang baru-baru ini diakses oleh prosesor dan umumnya dikenal sebagai cache primer. Selain itu, ini juga disebut sebagai cache internal atau cache sistem. Di antara berbagai jenis cache, cache L1 dianggap yang tercepat karena desainnya ini memiliki ukuran yang paling kecil dibandingkan dengan cache lainnya, biasanya L1 hanya berukuran puluhan *kilobyte*, namun ia menawarkan kecepatan tercepat dari semua jenis *cache*.

b) L2 (Level 2)

L2 cache adalah cache memori yang terletak pada chip prosesor yang sama tetapi di luar inti prosesor. Ini lebih lambat dari cache L1 tetapi mulai diintegrasikan ke dalam prosesor. L2 cache, juga dikenal sebagai cache sekunder atau eksternal, membantu menjembatani memori jarak jauh dan menyediakan data ke prosesor dengan cepat tanpa penundaan. Tujuan utamanya adalah untuk mengurangi waktu akses terhadap data yang telah diakses sebelumnya, sehingga tidak perlu ditransfer lagi ke prosesor.

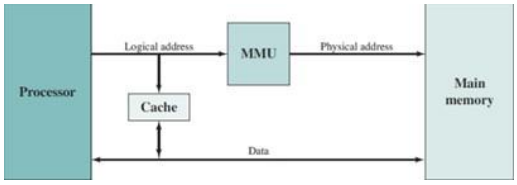
Cache L2 lebih besar dari cache L1, dengan ukuran mulai dari 64 kilobyte hingga ukuran lebih besar seperti 256 Kb, 512 Kb, atau 1024 Kb. Meskipun ukurannya lebih besar, cache L2 beroperasi pada kecepatan yang lebih lambat dibandingkan cache L1, dengan nilai latensi sekitar 2 hingga 10 kali lebih lambat. Cache L2 tidak wajib dan beberapa prosesor, terutama model lama sebelum Intel Pentium, mungkin tidak menyertakannya.

c) L3 (Level 3)

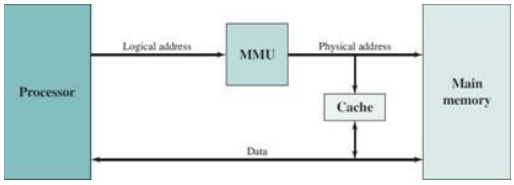
Dalam hierarki cache, L3 Cache memiliki kapasitas terbesar namun kecepatannya paling lambat dibandingkan cache lainnya. Ia bertindak sebagai perantara antara cache L2 dan L1, menyediakan data sebelum mencapai cache tercepat. Meskipun kecepatannya lebih lambat, cache L3 masih lebih cepat dibandingkan RAM komputer. Proses pencarian data prosesor dimulai dari cache L1, kemudian berpindah ke cache L2 jika diperlukan, dan terakhir ke cache L3. Jika data tidak ditemukan di salah satu cache, prosesor akan mengambilnya dari memori utama, khususnya RAM.

2) *Cache Address*

Kebanyakan prosesor dapat menggunakan memori virtual, yang membantu program mengelola memori berdasarkan aturan logis, bukan batasan fisik. *Memory Management Unit* (MMU) mengubah alamat virtual menjadi alamat fisik di memori utama untuk membaca dan menulis (Mufida et al., 2021). Cache dapat ditempatkan antara prosesor dan MMU atau antara MMU dan memori utama. Logical Cache, atau cache virtual, menyimpan data menggunakan alamat virtual dan memungkinkan kecepatan akses lebih cepat dibandingkan dengan cache fisik.



Gambar 3. 2 Logical Cache sumber ()



Gambar 3. 3 Pyhsical Cache

Salah satu kelemahan yang signifikan adalah bahwa dalam banyak sistem memori virtual, setiap aplikasi dialokasikan ruang alamat memori virtual yang sama, sehingga setiap aplikasi dimulai pada alamat 0. Hal ini dapat menyebabkan kebingungan karena alamat virtual yang sama dalam dua aplikasi berbeda mungkin sebenarnya berhubungan dengan alamat yang berbeda. alamat fisik.

3) *Cache Size*

Sebagaimana diuraikan dalam tabel dibawah, ukuran cache memainkan peran penting dalam mengkategorikan memori. Tujuannya agar cache berfungsi lebih cepat dibandingkan memori utama. Namun, permintaan akan kapasitas cache yang lebih besar menyebabkan jumlah gerbang yang dibutuhkan untuk mengelola cache menjadi lebih banyak, sehingga menyebabkan kinerja sedikit menurun dibandingkan dengan cache yang lebih kecil (Stallings, 2016). Tantangan ini diperparah dengan terbatasnya ruang *processor* dan *board*.

Processor	***Type***	***Year of Introduction***	**L1 *Cache***	**L2 *Cache***	**L3 *Cache***
PDP-11/70	*Minicomputer*	1975	1 kB	-	-
IBM 3090	*Mainframe*	1985	128–256 kB	-	-
Pentium	PC	1993	8 kB/8 kB	256–512 kB	-
Itanium 2	*PC/server*	2002	32 kB	256 kB	6 MB
IBM POWER5	*High-end server*	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	*Supercomputer*	2004	64 kB/64 kB	1 MB	-

Intel Core i7 EE 990	*Workstation/ server*	2011	6 * 32 kB/ 32kB	1.5 MB	12 MB

Tabel 3. 1 Cache size sumber (Stallings, 2016)

4) *Cache Mapping*

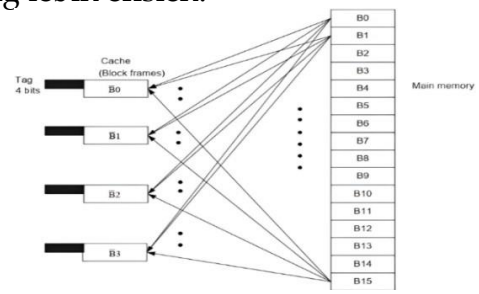
Untuk mengelola jumlah baris cache yang terbatas secara efisien dibandingkan dengan blok memori utama, penting untuk menerapkan algoritma untuk memetakan blok memori utama ke baris cache. Selain itu, diperlukan metode untuk mengidentifikasi blok memori utama mana yang sesuai dengan baris cache tertentu. Pemilihan fungsi pemetaan memainkan peran penting dalam membentuk organisasi cache. Tiga teknik pemetaan potensial meliputi :

a) *Direct Mapping* (Pemetaan langsung)

Memanfaatkan pendekatan langsung pemetaan langsung melibatkan penempatan setiap blok memori utama ke satu baris cache. Teknik ini tidak hanya efisien dan hemat biaya, namun juga mudah diterapkan. Namun, kelemahan metode ini adalah potensi penghancuran cache ketika beberapa blok dipetakan ke baris cache yang sama. Hal ini dapat menyebabkan inefisiensi dan menghambat kinerja sistem secara keseluruhan.

b) *Associatif mapping* (Pemetaan assosiatif)

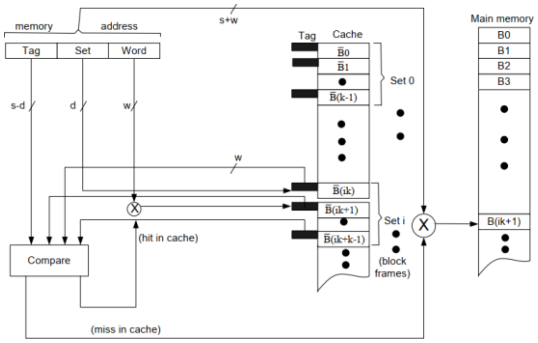
Pemetaan asosiatif memungkinkan banyak fleksibilitas dalam memetakan blok cache. Dengan metode ini, setiap blok di memori utama dapat ditempatkan di frame blok mana pun yang terbuka di cache. Pendekatan ini memberikan kebebasan kepada pengguna untuk memilih di mana dalam cache mereka ingin memetakan konten blok memori, sehingga menghasilkan penggunaan ruang dalam cache yang lebih efisien.



Gambar 3. 4 associatif mapping sumber ()

c) *Set associatif mapping*

Set associative menggabungkan efisiensi dari metode direction mapping dan fleksibilitas dari metode associative. Dengan mengatur blok cache ke dalam kumpulan dan menentukan aturan pemetaan, blok memori utama dapat dialokasikan ke kumpulan tertentu dalam cache. Desain ini memungkinkan peningkatan kinerja dan pengurangan waktu akses, karena cache disusun menjadi beberapa set, masing-masing berisi beberapa baris. Pendekatan canggih ini mengoptimalkan pengambilan memori dan meningkatkan efisiensi sistem secara keseluruhan.



Gambar 3. 5 K-way set associative mapping sumber ()

Dalam ilustrasi diatas ini menggambarkan teknik pemetaan yang dikenal sebagai k-way set associative mapping, di mana (m) dari block frame dipisahkan menjadi himpunan v=m/k, dengan setiap himpunan diidentifikasi dengan nomor himpunan d-bit, di mana 2d = v. Tag block cache kemudian menyusut menjadi s-d bit. Dalam aplikasi praktis, ukuran himpunan k biasanya berada dalam nilai 2, 4, 8, 16, atau 64. (Arie S.M & Lumenta, 2013).

3. *Main Memory* (Memori Utama)

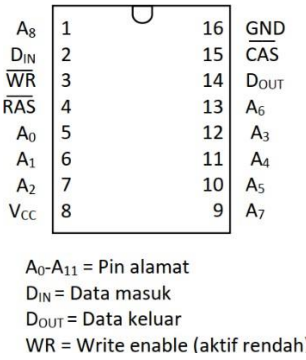
Memori utama terletak pada motherboard, menyimpan data untuk pemrosesan dan program yang digunakan untuk memproses data. Ia harus memiliki kapasitas yang cukup untuk menampung semua informasi yang diperlukan. Data disimpan dalam alamat tertentu agar mudah diakses. Penjelasan ini memperjelas detail memori internal, lokasinya, dan kapasitasnya.

1) *Random Access Memory* (RAM)

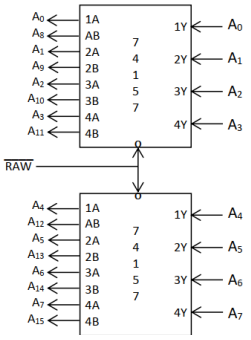
RAM (Random Access Memory), merupakan salah satu komponen kunci dalam komputer yang menyimpan sistem operasi, aplikasi, dan informasi prosesor. Hal ini memungkinkan prosesor mengakses data secara acak dan cepat. RAM sendiri bersifat volatil, yang artinya data akan hilang jika listrik padam, sehingga hanya cocok untuk penyimpanan sementara. Ram sendiri memiliki dua jenis ram yaitu; DRAM (Dynamic Random Access Memory), memiliki biaya per-bit yang lebih murah namun memerlukan dukungan ekstra untuk menyegarkan sel, dan satu jenis lainnya yaitu SRAM (Static random Access Memory), pada memori ini memiliki kecepatan akses yang lebih cepat dan biasanya digunakan untuk memori cache.

a) *Dynamic Random Access Memory* (DRAM)

RAM Dinamis, atau DRAM, seperti sistem penyimpanan yang menggunakan muatan kapasitor kecil untuk menyimpan informasi. Tagihan ini perlu diperbarui secara berkala untuk memastikan informasi tidak hilang. Sel DRAM hanya dapat menyimpan data dalam waktu singkat sebelum perlu diisi ulang. Istilah “dinamis” mengacu pada bagaimana muatan dapat bocor seiring berjalannya waktu. Chip DRAM TMS4464 dapat menampung data sebesar 256 KB dan memiliki pin serta saluran khusus untuk membantu mengakses dan menyimpan informasi secara efisien.



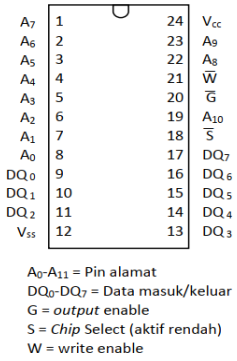
Gambar 3. 6 Pin-out DRAM TMS4464



Gambar 3. 7 Multipleks alamat untuk DRAM TMS4464

b) *Static Random Access Memory* (SRAM)

Di lain sisi, RAM Statis atau SRAM menggunakan komponen logika yang sama dengan yang ditemukan pada prosesor seperti perangkat digital. SRAM menyimpan nilai biner melalui pengaturan gerbang logika flip-flop konvensional. Memori jenis ini menampung data untuk sementara selama menerima daya, sehingga cocok untuk kebutuhan kapasitas memori yang kecil. SRAM TMS4016 adalah tipe khusus SRAM 2K x 8 Bit dengan 11 input alamat dan 8 pin input/output data. Diagram pada Gambar 8 menampilkan pin pada chip SRAM TMS4016.



Gambar 3. 8 Pin-Out SRAM TMS4016

2) ROM / PROM / EPROM / EEPROM

ROM (*Read Only Memory*) adalah memori non-volatile artinya tidak memerlukan sumber daya untuk memelihara data, ROM ini berisi data tetap dan tidak dapat ditulisi, biasanya digunakan untuk pemrograman mikro dan fungsi yang sering digunakan. Lalu PROM (*Programmable ROM*) adalah versi ROM yang dapat diprogram, menawarkan fleksibilitas dan kenyamanan dengan proses penulisan yang dilakukan secara elektrik. EPROM (*Erasable Programmable ROM*), dapat diprogram dan dihapus menggunakan cahaya dan listrik, memerlukan paparan sinar ultraviolet untuk menghapus informasi sebelum menulis data baru (Nur hayati, S.ST, 2017). Sedangkan EEPROM (*Electrically Erasable PROM*) adalah versi ROM yang lebih canggih, memungkinkan penghapusan listrik dan penulisan tanpa menghapus konten sebelumnya, menawarkan keunggulan non-volatilitas dan fleksibilitas meskipun prosesnya membutuhkan waktu lebih lama. Semua jenis

memori ini memiliki peran penting dalam penyimpanan dan pemrosesan data.

3) Flash Memory

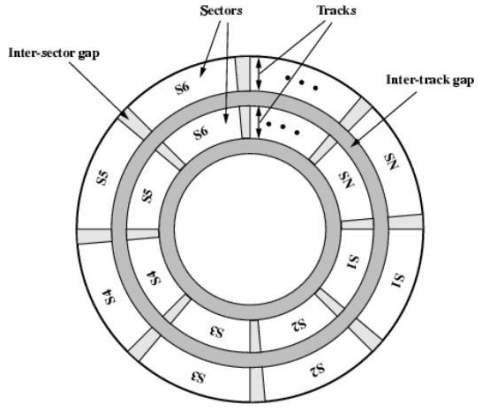
Jenis memori semikonduktor lainnya adalah memori flash, dinamai karena kecepatan pemrograman ulangnya yang cepat dan umumnya dikenal sebagai Read-Mostly Memory (RMM). Memori flash juga kadang-kadang disebut sebagai EEPROM (Electrically Erasable Programmable ROM), EAPROM (Electrically Alterable Programmable ROM), atau NVRAM (Non-Volatile RAM). Flash memiliki keunggulan non-volatilitas yang baik, dalam hal kemampuan penulisan ulang sistem dan densitas yang tinggi, dan telah menemukan banyak aplikasi. Flash berarti bahwa konten dari seluruh susunan memo, atau blok memory, dihapus dalam satu langkah (Jun, 2001).

4. *Eksternal Memory* (Memori Eksternal)

Memori eksternal ibarat tempat khusus komputer menyimpan informasi penting yang tidak akan hilang saat komputer dimatikan. Ini digunakan untuk menyimpan program dan data dalam waktu lama, seperti pada magnetic disk, optical disc, dan magnetic tape. Komputer dapat mengakses memori ini dengan perangkat khusus.

1) *Magnetic Disk*

Disk adalah benda berbentuk lingkaran yang terbuat dari logam atau plastik, dengan permukaan yang dapat dimagnetisasi dan dilapisi dengan bahan yang memungkinkan data dibaca atau ditulis menggunakan kumparan penghantar (head). Head tetap diam, sementara piringan berputar sesuai arah. Data pada disk ditata dengan dua jenis seperti yang diilustrasikan pada gambar (3.9): kecepatan sudut konstan dan perekaman beberapa zona. Data tersebut disusun dalam track-track yang dipisahkan oleh celah, dengan track-track tersebut berada dalam cincin konsentris.



Gambar 3. 9 Magnetic Disk sumber ()

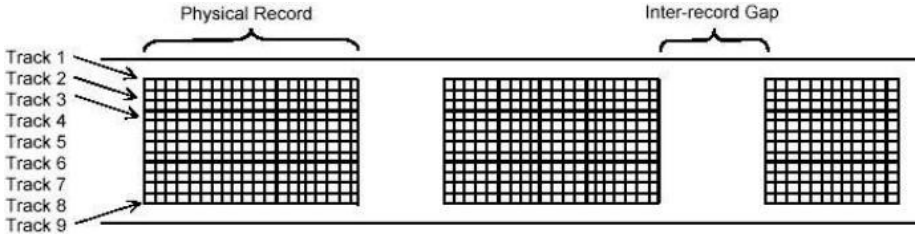
Fungsi gap membantu mencegah kesalahan saat membaca dan menulis informasi pada disk komputer. Gap memastikan bahwa data disimpan di tempat yang tepat dan dapat ditemukan dengan mudah. Disk dibagi menjadi beberapa track, dengan setiap track menyimpan sejumlah data tertentu. Data disusun dalam blok-blok, yang merupakan bagian-bagian kecil dalam setiap jalur. Disk drive menggunakan penanda khusus untuk mengetahui di mana setiap blok dimulai dan berakhir. Penanda ini membantu disk drive menemukan informasi yang tepat dengan cepat.

2) *Optical Disk*

Optical disk adalah medium penyimpanan elektronik yang dapat ditulis dan dibaca menggunakan sinar laser berkekuatan rendah. Optical Disk ini dapat menampung data mulai dari kisaran 550 Mb - 800MB. Optical Disk sendiri memiliki berbagai produk seperti CD (*Compact Disk*) , CD-ROM (*Compact Disk Read Only Memory*), CD-I (*Compact Disk Interactive*), DVI (*Digital Video Interactive*), WORM (*Write One Read Many*), dan *Erasable Optik Disk* (Victor Amrizal & Qurrotul Aini, 2013).

3) *Magnetic Tape*

Sistem pita magnetik dan sistem disk magnetik menggunakan teknik membaca dan menulis yang sama. Di masa lalu, sistem pita magnetik memiliki 9 jalur untuk menyimpan data, dengan satu byte dan satu bit paritas pada setiap jalur. Sistem tape modern memiliki 18 atau 36 track untuk mendukung lebar kata digital. Mirip dengan disk, pita magnetik dibaca dan ditulis dalam blok kontinu yang dikenal sebagai rekaman fisik, dipisahkan oleh celah antar rekaman. Tata letak fisik pita magnetik ditunjukkan pada Gambar 6.18.



Gambar 3. 10 Magnetic Tape sumber ()

Kepala pita magnetik berfungsi sebagai perangkat akses berurutan, yang memerlukan penyesuaian untuk mencari dan membaca atau menulis catatan. Jika head diposisikan di atas rekaman yang diinginkan, pita harus dibalik sebelum melanjutkan ke arah depan. Cara ini berbeda dengan teknologi disk yang memanfaatkan metode akses langsung. Karena kecepatan putarannya yang lambat dan kecepatan transfer data yang lamban, pita magnetik secara bertahap mulai ditinggalkan dan digantikan oleh produk CD. (Ir. Heru Nurwarsito, 2012)

4. KESIMPULAN

hierarki memori dalam arsitektur komputer memiliki peran penting dalam meningkatkan kinerja sistem secara keseluruhan. Berbagai jenis memori, mulai dari register hingga memori sekunder seperti magnetic disk dan optical disk, memiliki kelebihan dan kekurangan masing-masing dalam hal kecepatan, kapasitas, dan biaya per bit. Penelitian ini menyoroti pentingnya pemahaman mendalam tentang struktur memori untuk merancang sistem memori yang optimal sesuai dengan kebutuhan komputasi modern. Selain itu, teknik pemetaan seperti *k-way set associative mapping* juga menjadi faktor penting dalam mengoptimalkan pengambilan memori dan efisiensi sistem secara keseluruhan. Dengan demikian, pemahaman yang baik tentang hierarki memori dan berbagai jenis memori dapat membantu dalam meningkatkan performa dan efisiensi sistem komputer secara keseluruhan.

5. IMPLEMESTASI SIMULASI SEDERHANA

```python
class CacheMemory:
    def __init__(self, size):
        # Ukuran cache memory
        self.cache = []
        self.size = size

    def access_data(self, data):
        # Mengecek apakah data sudah ada di dalam cache
        if data in self.cache:
            print(f"Data {data} sudah ada dalam cache.")
            # Jika data ada, hapus data tersebut dan masukkan lagi di akhir untuk menandakan bahwa data ini baru digunakan
            self.cache.remove(data)
            self.cache.append(data)
        else:
            print(f"Data {data} tidak ada dalam cache.")
            # Jika cache penuh, hapus data yang paling lama digunakan (LRU)
            if len(self.cache) >= self.size:
                removed_data = self.cache.pop(0)
                print(f"Cache penuh. Menghapus data {removed_data} dari cache.")
            # Masukkan data baru ke dalam cache
            self.cache.append(data)

        # Menampilkan kondisi cache setelah akses
        print("Cache sekarang:", self.cache)

# Simulasi dengan kapasitas cache 3 blok
cache_memory = CacheMemory(3)

# Akses beberapa data
cache_memory.access_data(1)
cache_memory.access_data(2)
cache_memory.access_data(3)
cache_memory.access_data(4)
cache_memory.access_data(1)
cache_memory.access_data(5)
```

```
Data 1 tidak ada dalam cache.
Cache sekarang: [1]
Data 2 tidak ada dalam cache.
Cache sekarang: [1, 2]
Data 3 tidak ada dalam cache.
Cache sekarang: [1, 2, 3]
Data 4 tidak ada dalam cache.
Cache penuh. Menghapus data 1 dari cache.
Cache sekarang: [2, 3, 4]
Data 1 tidak ada dalam cache.
Cache penuh. Menghapus data 2 dari cache.
Cache sekarang: [3, 4, 1]
Data 5 tidak ada dalam cache.
Cache penuh. Menghapus data 3 dari cache.
Cache sekarang: [4, 1, 5]
```

Implementasi simulasi sederhana ini bertujuan untuk menggambarkan bagaimana algoritma Least Recently Used (LRU) bekerja dalam sistem cache memory. Dalam simulasi ini, dibuat sebuah kelas CacheMemory yang merepresentasikan cache dengan ukuran tetap, yaitu tiga blok. Setiap kali data diakses, sistem akan memeriksa apakah data tersebut sudah ada di dalam cache. Jika sudah ada, data tersebut dianggap baru digunakan kembali, sehingga dihapus dari posisinya lalu dimasukkan kembali ke akhir cache. Namun, jika data belum ada dan cache belum penuh, data langsung dimasukkan. Jika cache sudah penuh, maka data yang paling lama tidak digunakan (yang berada di posisi paling awal) akan dihapus, dan data baru dimasukkan. Proses ini menunjukkan prinsip kerja LRU, di mana sistem selalu menggantikan data yang paling jarang atau paling lama tidak diakses. Hasil simulasi menunjukkan perubahan isi cache setelah setiap akses data, memperlihatkan bagaimana LRU menjaga agar data yang paling relevan tetap tersedia dalam cache. Simulasi ini bermanfaat sebagai alat pembelajaran untuk memahami manajemen memori dalam arsitektur komputer secara praktis dan visual.

REFERENCES

Abdurohman, M. (2017). *Organisasi dan Arsitektur Komputer : edisi Revisi Keempat* (4th ed.). INFORMATIKA.

Arie S.M, & Lumenta. (2013). Organisasi Cache Memory. *Teknik Elektro Dan Komputer*, 1–5.

Ir. Heru Nurwarsito, M. K. (2012). *Arsitektur dan Organisasi : Komputer Ekstenal*. UB Distance Learning.

Jun, Z. (2001). *Flash memory technology development*. *1*(100084), 189–194. https://doi.org/10.1109/ICSICT.2001.981453.

Mufida, E., Leidiyana, H., Rachmawati, E., & Hertyana, H. (2021). *Arsitektur Komputer Struktur dan Fungsi*. 1–221.

Munazilin, A. (2017). *Arsitektur Komputer* (1st ed.). CV. BUDI UTAMA.

Nur hayati, S.ST, M. (2017). *DIKTAT KULIAH : ORGANISASI DAN ARSITEKTUR KOMPUTER TEU2211*. Fakultas Teknik UMY. https://medium.com/@arifwicaksanaa/pengertian-use-case-a7e576e1b6bf

Putri, E. D. (2008). *ALGORITMA DIVIDE AND CONQUER UNTUK MEMBUAT KERJA CACHE PADA KOMPUTER PARALEL LEBIH EFISIEN*. 4.

Rima Rizqi Wijayanti, S.ST., M. M. S. (2023). *Arsitektur dan Organisasi Komputer.* (P. T. Cahyono (ed.)). CV. Rey Media Grafika.

Stallings, W. (2016). *Computer Organization and Architecture 10th Edition*. Pearson Education, Inc.

Suryawinata, M. K. (2018). *Buku Ajar : Arsitektur dan Organisasi Komputer* (M. P. Septi Budi Sartika (ed.); Vol. 1, Issue 1). UMSIDA PRESS.

Victor Amrizal & Qurrotul Aini. (2013). Arsitektur Komputer : Teori dan Perkembangannya. In *Arsitektur Komputer teori dan perkembangannya* (Issue 2). Halaman Moeka Publishing.